A SEMANTIC INDEXING OF ELECTRONIC DOCUMENTS IN OPEN FORMATS

Vyacheslav Bessonov, Viacheslav Lanin, George Sokolov

Abstract*: Currently many issues related to the development of semantic indexing methods remain open. The problem of search pertinence increasing with a low time-complexity is one of the major research issues in Computer Science. Semantic search as an alternative solution to this problem has a high time complexity. This paper describes the use of agent-based approach to reduce the time complexity of constructing semantic indexes used for searching. Obvious, once created index must be stored in document. The second part of this paper describes an approach for inclusion of metadata in electronic documents which have such open document format as OpenXML or Open Document Format. The proposed approach enables to perform an effective semantic indexing and the semantic searching. Semantic indexing of electronic documents is intended to include special structure associated with the content of documents in its metadata. Most of the currently used electronic document formats do not permit the inclusion of additional information, on the contrary electronic documents open formats Office Open XML and OpenDocument Format allow to solve this problem.*

Keywords*: Semantic indexing; Agent; Ontology; Document, Open XML; OpenDocument Format; Metadata.*

ACM Classification Keywords*: I.2 ARTIFICIAL INTELLIGENCE: I.2.11 Distributed Artificial Intelligence; I.7 DOCUMENT AND TEXT PROCESSING: I.7.2 Document Preparation; I.7.3 Index Generation.*

Introduction

Nowadays the information retrieval (from the Internet and off-line sources) is one of the major research areas in Computer Science. The main criteria of a successful search are the high relevance of search query information and fast response time. Traditional search engines typically use an approach «Bag of words» based on statistical methods to search for information. This approach takes precedence over semantic search methods is due to low time-complexity, low implementation complexity and satisfactory degree of relevance. One of the main areas of modern researches in the information retrieval is an increasing of search pertinence with a low time-complexity.

In syntactic search some indexes are built to find quickly the information required on some key words. By analogy let's introduce a concept of a semantic index. In this paper the semantic index is one-one correspondence between elements of the text and concepts from some ontological resource. There are different formats of the semantic indexes. Some of them are primitive (such as microformats hCard, Geo, microdata html5) and other formats are advanced (such as RDF, OWL, underlying the Semantic Web). In the semantic indexing there are two directions: the construction of semantic indexes and search for information on a semantic index. In this paper we will consider the construction of the semantic index (or the so-called semantic markup) for electronic documents.

The main problems of constructing semantic indexes are

1) high time-complexity (is due to various kinds of ambiguity that require paying respect of a context),

2) the problem of choosing ontology, which would be sufficiently complete to satisfy all search queries in an electronic document,

3) large amount of constructed semantic indexes and the problem of storage.

In this paper, the authors offer one approach of solving the first problem (the problem of time-complexity). Obviously, increase in the rate of the semantic indexing operation is required not one but several calculators, i.e. the parallelization of this operation is needed. The execution of the semantic markup operation requires the coordination of actions to resolve ambiguities. That's why simple asynchronous calculators aren't capable to solve the problem. According to the authors the most appropriate solution is using agent-based approach.

Existing Approaches

Solution to the agent-based semantic indexing problem can be obtained in two ways:

1) using of generic agent-based platforms that can decide a wide range of tasks,

2) using of specialized semantic indexing systems based on the multi-agent paradigm.

Let us consider each of these methods. Most popular agent platforms are JADE [1], MASDK [2], Zeus [3].

Table 1. Generic agent-based platforms

	JADE	MASDK	ZEUS
Developer community	Telecom Italia Lab	SPIIRAS	BT Laboratories
License	LGPL	LGPL	LGPL
Description	This is the platform for rapid development of multi-agent systems, which implements FIPA standards [4]. JADE provides base classes for creating agents and infrastructure for the operation of multi-agent system.	This is the software environment for multi-agent application development that supports the full life cycle application development of MAS. The agent platform, which is the part of MASDK, works on the principle of P2P.	This is the agent platform designed for rapid development of multi-agent applications. Zeus provides a library of agent components.
Description of the agent behavior	Set in the code of the agent class that inherits from Agent.	Set with language ASML. This language is used for generating applied MAS.	Set in an environment for building agents, from which the agent code is generated.

Each of these agent platforms allows one way or another to describe the behavior of the agent. Depending on the platform we can define almost any behavior of an agent, programming or describing it using specific language. So we can determine the behavior of the agent that implements mechanisms of semantic indexing. The key problem of this approach is the high overhead of run-time. This is due to a complex infrastructure applications received applications. This can be compared with a programming in high level language and Assembler. The actions are the same, but the performance is significantly different. Therefore, such an approach to the problem is not satisfactory.

As noted above, the second approach to the problem of semantic indexing is the use of specialized semantic indexing systems based on the multi-agent paradigm. In this area, it was found only one solution – Magenta Toolkit [5]. This software solution is commercial, so there is no legal possibility to evaluate the effectiveness of work and, especially, to study the mechanisms of their internal functioning. Magenta Toolkit developers have written a number of publications [6, 7], which describe the principles of the system in outline without specifics. This decision is also not satisfactory.

Therefore, the task of the research is development of an open (open source and detailed descriptions of the principles) and an effective method of semantic indexing based on the multi-agents paradigm. In addition, you also need the option to apply this method to all electronic records. So the agent platform must be developed.

Document Analyses Steps

On Fig. 1 text mining process steps are shown. Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar.

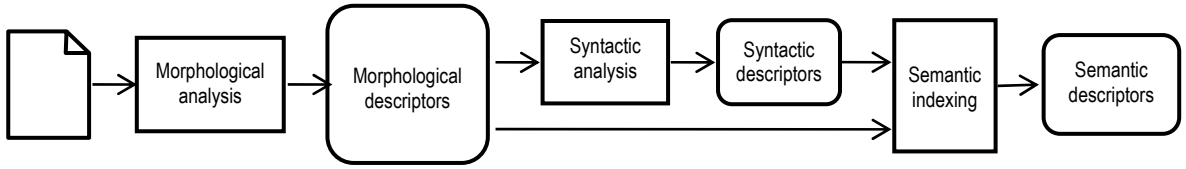


Figure 1. Steps of document analyses

Simplifying the problem we assume that first two steps of text mining process have been made, i.e. a set of syntactic and morphological descriptors for each sentence have been obtained. The result of semantic analysis (indexing) is a semantic descriptor of text that binds the syntactic descriptors of sentences to the elements of the domain ontology which is used for semantic search.

Descriptors (morphological, syntactic, and semantic) are a set of tags which marks words in the sentence. Syntactic and morphological descriptors will be put into relational tables for two reasons. Firstly, syntactic and morphological descriptors will be actively used for semantic indexing. Secondly, we don't want to pile up document by tags. Each word in the text (except for a different kind of stop words) will be assigned a unique identifier. Each identifier corresponds to a separate table row.

Thus, i-th row of the table looks like $(id_i, \{a_j\}_i)$, where id_i – the identifier of the word, $\{a_j\}_i$ – set of attributes (tags) that have been assigned to a given word during morphological and syntactic analysis process. In each row of syntactic descriptor table an identifier of applicable syntactic rule is indicated. The syntactic rule is a rule for constructing syntactically correct sentences. The semantic descriptor is represented as set of tags (semantic markup) within the indexed document.

Agent-based Solution

Further let us consider the process of building a semantic based on multi-agent approach (see Fig. 2).

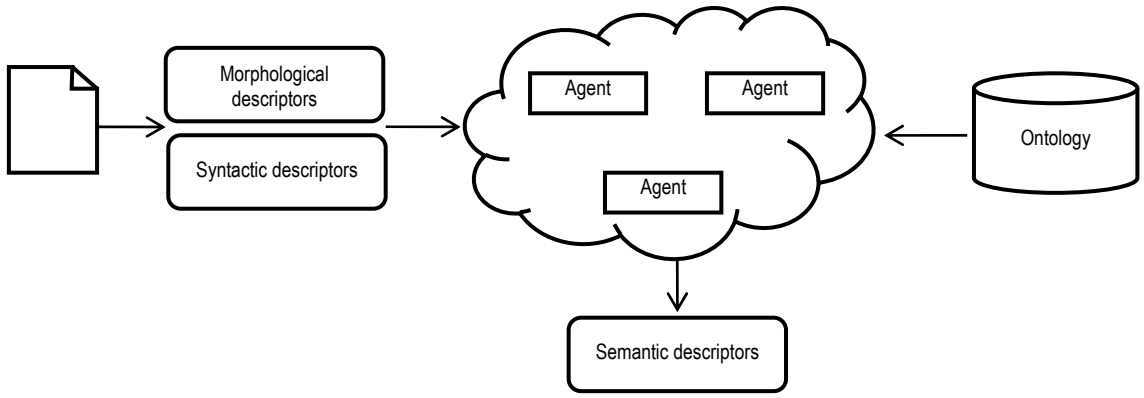


Figure 2. Architecture of agent platform

Each agent will have access to a common ontology, syntactic, morphological descriptors and electronic documents which will be indexed. Then analysis will be produced on the sentences in the text. Each agent is attached to a particular word (agent worker) in a sentence, for which there is a description in the descriptor tables (morphological and syntactic). Sentences are processed sequentially by agents. The agents form a "team" to index the particular sentence. If the number of words in a sentence greater than the number of active agents then two options could be: the agent of the analyzed sentence takes a few words, vacant agent of another sentence helps to analyze sentence. Thus, agents in the system after the start of the indexing are divided into teams. The number of agents in teams depends on the structure and content of the sentences in the document. Each team has a team leader agent. The team leader agent determines when the indexing of a sentence is completed and next sentence could be indexed. Besides the team leader agent resolves various kinds of the ambiguities by the auction.

Let us consider two levels of supervisors: team leader agent first-level, team leader agent second level. Team leader agent first-level aims to index all the sentences in the document. Team leader agent second level has a goal to index a specific sentence. Agent-worker has a goal to index a particular word in a sentence.

Agents are constantly involved not only in the form of communication "team leader agent – agent worker", but also in communications "agent worker – agent-worker" within the team. In addition, supervisors of different teams communicate with each other for the redistribution of available resources.

XML Based Document Format

Office Open XML (OOXML) is a set of open formats based on ZIP and XML technologies intended for representation of electronic documents package of office applications such as spreadsheets, presentations, text documents.

In 2006 the Office Open XML was recognized as the standard ECMA-376 and 2008 as the international standard ISO/IEC 29500:2008.

Since 2007 version of Microsoft Office OOXML is the default format for all applications included in the package of Microsoft Office.

For each document type its own markup language is used:

- WordprocessingML for text documents;
- SpreadsheetML for spreadsheets;
- PresentationML for presentations.

OOXML also includes a set of specialized markup languages that can be used in documents of various types:

- Office Math Markup Language is used to represent mathematical formulas;
- DrawingML is used to represent vector graphics and diagrams.

Office Open XML uses Open Packaging Convention (OPC), created by Microsoft and intended for storing a combination of XML and binary files (eg, BMP, PNG, AVI and etc.) in a single container file.

OpenDocument Format (ODF) is an open document file format intended for storing and exchanging editable office documents such as spreadsheets, text documents and presentations.

ODF standard is created and supported by Committee ODF Technical Committee organization OASIS (Organization for the Advancement of Structured Information Standards). OASIS published ODF 1.0 in May 2005, Commission International Organization for Standardization / International Electrotechnical Commission ratified it in May 2006 as ISO/IEC 26300:2006, so ODF become the first international standard for office documents.

ODF was accepted as the national standard in the Russian Federation, Brazil, Croatia, Italy, Korea, South Africa, Sweden and Venezuela.

Although both formats are based on open technologies, and are actually ZIP-archives that contain a set of XML-files defining the contents of the documents, they use very different approaches to solve the same problems and have radically different internal representation.

Format ODF reuses existing open XML standards, and introduces new ones only if it is really necessary. For example, ODF uses a subset of Dublin Core to represent document metadata, MathML to present mathematical expressions, SMIL to present multimedia content of the document, XLink to provide hyperlinks, etc. It means primarily it is easy to use this format by people already familiar with the existing methods to process XML.

The Office Open XML Format uses solutions developed by Microsoft to solve these problems, such as, Office Math Markup Language, DrawingML, etc.

OFFICE OPEN XML AND OPENDOCUMENT FORMAT APIS

As mentioned above, despite the same set of used technologies – XML and ZIP, Office Open XML Format and the OpenDocument Format have very different internal representation. Besides over the formats are under permanent development, there are currently several revisions of each format with very different possibilities.

For the Office Open XML they are:

- ECMA-376;
- ISO / IEC 29500:2008 Transitional;
- ISO / EC 29500:2008 Strict.

For the OpenDocument Format they are:

- ISO / IEC 26300;
- OASIS ODF 1.1;
- OASIS ODF 1.2.

Existing software solutions designed to work with this formats are quite different. We will consider some of them.

All libraries and other software tools for working with documents in the Office Open XML Formats can be divided into two broad categories. We will reference these technologies next way:

- OPC API – low-level API, allowing to work with OPC-structure of OOXML documents, but do not provide opportunities to work with markup languages Office Open XML. Examples of those APIs are shown in Table 2.

Table 2. OPC APIs comparison

	ECMA-376	ISO/IEC 29500:2008	ISO/EC 29500:2008 Strict
Packaging API	+		
System.IO.Packaging	+		
OpenXML4j	+		
libOPC		+	

- OOXML API – high-level API, designed to work with specific markup languages (WordprocessingML, SpreadsheetML, PresentationML). Libraries and tools of this category typically are based on OPC API. Examples of OOXML APIs are shown in Table 3.

Table 3. OOXML APIs comparison

	ECMA-376	ISO/IEC 29500:2008	ISO/EC 29500:2008 Strict
Microsoft Office 2007 Automation		+	
Microsoft Office 2010 Automation		+	
Open XML SDK 2.0	+		
Apache POI		+	

Libraries for operating with electronic documents in the ODF format can be divided into two broad categories too:

- Libraries in the ODF Toolkit. ODF Toolkit Union is the community of open source software developers. Its goal is simplifying document and document content software management.

- Third-party organizations libraries.

Table 4. ODF APIs comparison

	ISO/IEC 26300	OASIS ODF 1.2
AODL		
odf4j	+	
ODFDOM	+	+
Simple Java for ODF		
lpOD	+	

SemanticLib Project

It is obvious that there should be a universal approach, allowed to work with electronic documents in various formats in a standardized way. Library SemanticLib was developed to solve this problem. The library provides a unified API for working with documents in two formats: Office Open XML and OpenDocument Format. The core library contains an abstract model of an electronic document that is a generalization of the models used in the Office Open XML and OpenDocument Format. Schematic representation of the model is shown in Fig. 3.

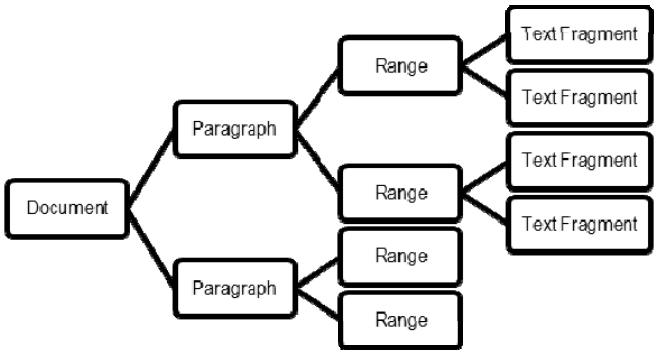


Figure 3. The model of the document used in SemanticLib

Fig. 4 shows a software implementation of DOM SemanticLib.

IMarkupable interface contains properties and methods that are used for semantic markup.

ITextDocument interface contains methods and properties for working with text documents, presented in a format like OOXML, and in the format ODF.

IParagraph interface contains properties and methods for working with particular paragraphs of the document.

IRange interface is used for working areas with continuous text contained in paragraphs.

IText interface is designed to work with particular text fragments contained in the text fields. The reason for the separation is the necessary to provide an opportunity for semantic markup of particular words in a text document.

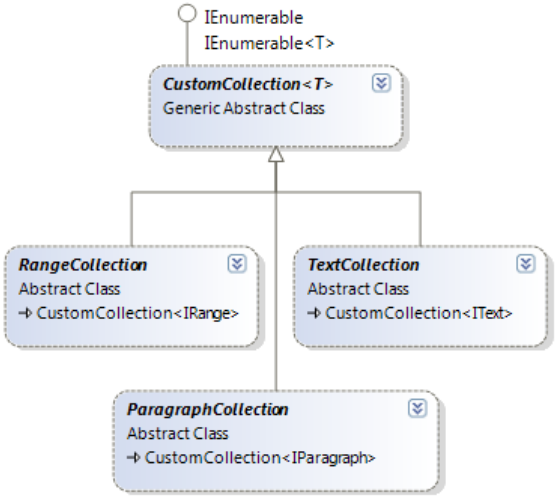


Figure 4. SemanticLib.Core.dll interfaces to work with OOXML and ODF documents

It is worth to note that all mentioned interfaces inherit from interface IMarkupable, so the semantic markup can be used as well as at the level of the document and to its particular elements such as paragraphs, text fields and text fragments.

It was mentioned that a text document and its fragments are containers, i.e. they contain other elements:

- a text document contains a collection of paragraphs;
- each section contains a collection of text fields;
- each text area contains a collection of text fragments.

Fig. 5 shows the hierarchy of abstract classes that represent collections of text documents.

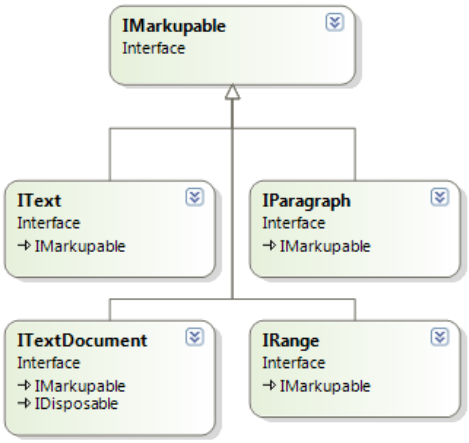


Figure 5. Collections of DOM SemanticLib

CustomCollection is the base class for all collections *SemanticLib*. It contains the common set of properties and methods, such as, for example, adding a new item in a collection, inserting a new item in a collection, removal element of the collection, etc. *ParagraphCollection* represents a collection of paragraphs. *RangeCollection* represents a collection of text fields. *TextCollection* represents a collection of text fragments.

SEMANTICLIB PLUG-IN SYSTEM

In SemanticLib implementation plug-ins based architecture is used. The library core contains only a description of the document model (DOM) and implementation of the methods for processing documents of any format is contained in the plug-ins. Typically each plug-in is an implementation of SemanticLib DOM with some libraries described in paragraph V. For example, a plug-in SemanticLib.OpenXmlSdkPlugin.dll uses API Open XML SDK 2.0, a plug-in SemanticLib.LibOpcPlugin.dll contains API libOPC.

Using plug-ins using makes possible a high degree of flexibility and extensibility. If a library expire or a new one appears, developer can just replace or add a plug-in without changing the basic functions of libraries and existing code.

However, plug-in development becomes significant difficult because of the existing the variety and diversity libraries. For example, the library Office Open XML SDK 2.0 is created on the platform .NET, while the library ODFDOM is created in Java, which means a significant difficulty trying to promote interoperability between these libraries. It is also difficult to ensure interoperability between C/C++ and .NET libraries. Let's consider how these issues are resolved in SemanticLib.

Let's see the interoperability between C/C++ and .NET code by the example LibOpcPlugin, which is the implementation of DOM SemanticLib with libraries libOPC, written in ANSI C.

It was decided to use C++/CLI to enable interoperability between managed and unmanaged code. The main advantage of this solution is the ability to use object-oriented programming style even interacting with procedural code of libOPC. In this case plug-in consists of a set of classes that implement the interfaces of DOM SemanticLib.

Interoperability between Java and .NET code will be considered on the example plug-SemanticLib.OdfDomPlugin.

There are some solutions to ensure interaction between Java and .NET applications. For example, there are products of JNBridge company, which provide both in-process and inter-process (network cloud) communication.

However, in SemanticLib Open Source project jni4net was selected. Its aim is providing an in-process communication.

Deal with jni4net has several stages:

- Creating a proxy for a Java library with a special utility proxygen, which is part of jni4net.
- Creating a .NET stub, which provides the work with Java-proxy. This step is also performed using proxygen.
- Implementation of a plugin functional using the resulting stub.

This process is quite complex and requires specific skills, so it is necessary to create automation tools in future versions of SemanticLib.

One of the significant advantages offered by SemanticLib, is the ability to work with a dynamic plug-ins. This feature is important if you work with a large number of different plug-ins. Plug-in manager, which is part of SemanticLib, helps user to manage the plug-ins loading process.

Plugin Manager provides the following features:

- Find the required plug-ins in accordance with certain criteria, such as the name of the plug-in or the format of the document.
- Loading and unloading plug-ins.
- Viewing the meta-information about the loaded plug-ins (name, manufacturer, document format, etc.).

Conclusion

So, in this paper we have discussed various approaches to solving the problem of document semantic indexing based on multi-agent paradigm. We propose a variant of the solution of that problem and describe it in terms of morphological, syntactic and semantic descriptors of the text. Specialized types of agents are introduced and the general principle of multi-agent system is described.

Semantic indexing of documents in Open XML Formats and Open Document Format can be implemented on the basis of the described solutions. The developed library is a part of the intelligent document processing project, but also can be used to solve other problems that require metadata inclusion.

Acknowledgement

The paper is published with financial support by the project ITHEA XXI of the Institute of Information Theories and Applications FOI ITHEA (www.ithea.org) and the Association of Developers and Users of Intelligent Systems ADUIS Ukraine (www.aduis.com.ua).

Bibliography

[1] JADE Programmers guide. http://sharon.cselt.it/projects/jade /doc/programmersguide.pdf
[2] Gorodetsky V., Karsan O., Samoilov V., Serebryakov S. "Applied multi-agent systems of group control" / / Artificial intelligence and decision making № 2.2009
[3] ZEUS Techical Manual. www.upv.es/sma/plataformas/zeus/Zeus-TechManual.pdf
[4] The Foundation for Intelligent Physical Agents. http://www.fipa.org
[5] The Magenta Toolkit. http://www.magenta-technology.ru/ru/
[6] Andreev V., Iwkushkin K., Karyagin D., Minakov I., Rzevski G., Skobelev, P., Tomin M.: Development of the Multi-Agent System for Text Understanding. In 3rd International Conference 'Complex Systems: Control and Modelling Problems'. Samara, Russia, September 4-9, 2001, 489-495.
[7] Minakov I., Rzevski G., Skobelev, P., Kanteev M., Volman S.: Multi-Agent Meta-Search Engine Based on Domain Ontology. http://www.magenta-technology.ru/ru/
[8] ISO/IEC 29500-1 Second edition, 2011-08-15. Information technology – Document description and processing languages – Office Open XML File Formats. Part 1: Fundamentals and Markup Language Reference.
[9] ECMA-376 Third Edition, June 2011. Office Open XML File Formats – Part 1.
[10] ISO/IEC 26300 First edition, 2006-12-01. Information technology – Open Document Format for Office Applications (OpenDocument) v1.0.
[11] Open Document Format for Office Applications (OpenDocument) Version 1.2 Part 1: OpenDocument Schema 29 September 2011.
[12] Open Document Format for Office Applications (OpenDocument) Version 1.2 Part 3: Packages 29 September 2011.
[13] OASIS OpenDocument 1.2 File Format Change Proposal Metadata Model, 6 Jun 2007.
[14] OASIS OpenDocument 1.2 Metadata Examples, Oct 2, 2009.
[15] Wouter van Vugt. Open XML. The markup explained.

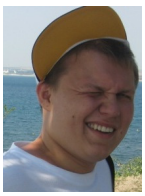
Authors' Information

Vyacheslav Bessonov *– Perm State University, Department of Software Computing; Russia, Perm, 614990, Bukireva St.,15; e-mail: v.bessonov@hotmail.com.*

Major Fields of Scientific Research: Software technologies, NLP.

Viacheslav Lanin *– Perm State University, Department of Software Computing; Russia, Perm, 614990, Bukireva St.,15; e-mail: lanin@perm.ru.*

Major Fields of Scientific Research: Intelligent agents, Ontologies, Document processing.



George Sokolov – *Perm State University, Department of Software Computing; Russia, Perm, 614990, Bukireva St.,15; e-mail: sokolovgeorge@gmail.com.*

Major Fields of Scientific Research: Intelligent agents, Semantic indexing, Graph drawing.